STUDIA INFORMATICA 2011
Volume 32 Number 2A (96)

Agnieszka KOMOROWSKA
Politechnika Warszawska, Instytut Informatyki

ZASTOSOWANIE REGUŁ ASOCJACYJNYCH W ANALIZIE DANYCH Z SYMULATORÓW BŁĘDÓW

Streszczenie. Artykuł opisuje metodykę analizy danych z symulatora błędów za pomocą reguł asocjacyjnych, przedstawioną na przykładzie analizy wyników eksperymentów symulacji błędów dla sterownika reaktora chemicznego. Dane z eksperymentów gromadzone są w hurtowni danych, wyposażonej w usługi raportowania i eksploracji danych. Celem analizy jest odkrycie cech aplikacji istotnych dla jej niezawodności.

Słowa kluczowe: odkrywanie wiedzy, eksploracja danych, reguły asocjacyjne, symulacja błędów, programowe wstrzykiwanie błędów

APLICATION OF ASSOCIATION RULES IN ANALISIS OF DATA FROM FAULTS SIMULATORS

Summary. Article describes a methodology of analysis of data from faults simulator. The methodology is based on association rules and is presented on an example of analysis of results from faults simulation for chemical reactor controller. The goal of the analysis is to discover application's features important for its dependability.

Keywords: data mining, data exploration, associations rules, faults simulation, software fault injection

1. Wstęp

Symulatory błędów dają możliwość wielokrotnego sprawdzenia zachowania się badanej aplikacji w sytuacji wyjątkowej i wyposażenia jej w odpowiednie mechanizmy wykrywania i tolerowania błędów [1, 2]. DInjector [8] jest przykładem rozproszonego system testowania odporności aplikacji na zakłócenia zasobów sprzętowych metodą programowego wstrzykiwania błędów. Ponieważ symulator zbiera szczegółowe dane o zachowaniu badanej aplikacji po

wstrzyknięciu błędu, wynikiem każdego eksperymentu jest duży wolumen danych, który trudno analizować bez wsparcie specjalistycznych narzędzi.

W tej sytuacji z pomocą przychodzą metody eksploracji danych. W pracy [6] opisano wykorzystanie drzew decyzyjnych do przetwarzania danych pochodzących z symulacji błędów w transakcyjnych bazach danych, przeprowadzonych w środowisku DBench-OLTP. Modele eksploracji danych efektywnie wsparły wyznaczenie czynników wpływających na niezawodność badanego systemu. Drzewa decyzyjne zostały także wykorzystane w Systemie Odkrywania Wiedzy w hurtowni wyników Eksperymentów Symulacyjnych (SOWES) [7]. Ewaluacje tego systemu przeprowadzono na danych pochodzących z programowego symulatora błędów FITS [2]. W tym przypadku na podstawie drzew decyzyjnych autorzy przeprowadzają analizę czynników wpływających na wynik testu.

Także DInjector posiada dedykowany systemem analityczny – FEARS. W pracy [4] opisano hurtownie danych oraz zestaw raportów z danych relacyjnych i wielowymiarowych. Niniejszy artykuł zawiera propozycję nowej metodyki analizy danych zgromadzonych w FEARS z wykorzystaniem metod odkrywania wiedzy. Pierwszym algorytmem, który jest dostępny w systemie FEARS są reguły asocjacyjne. Jego zaletą jest łatwe przełożenie modelu analitycznego na relacje ze świata rzeczywistego. Algorytm ten dał również pozytywne wyniki w podobnym zadaniu: reguły asocjacyjne w połączeniu z regresja liniową były wykorzystywane do poszukiwania przyczyn błędów aplikacji [5].

Rozdział 2 niniejszego artykułu opisuje wstrzykiwacz błędów DInjector. W rozdziale 3 przedstawiono krótko hurtownię danych FEARS. Rozdział 4 opisuje proces budowy modelu analitycznego na danych z hurtowni, rozdział 5 zaś zastosowanie opisanej metodyki do danych pochodzących z eksperymentów wstrzykiwania błędów w aplikacje sterownika rektora chemicznego. Podsumowanie i przyszłe kierunki badań zamieszczono w rozdziale 6.

2. Programowy wstrzykiwacz błędów – DInjector

Symulator błędów DInjector [8] pozwala badać zachowanie aplikacji w przypadku zakłóceń sprzętowych, na przykład przekłamanie pojedynczego bitu w rejestrze procesora lub pamięci komputera. Jest to rozbudowane, rozproszone środowisko, pozwalające na testowanie odporności aplikacji na błędy oraz weryfikację zastosowanych w nich mechanizmów wykrywania i tolerowania błędów.

DInjector pozwala na zaplanowanie i równoległe wykonanie na wielu maszynach eksperymentu składającego się z wielu testów. Rozdzielenie symulacji na różne maszyny pozwala na skrócenie czasu potrzebnego na przeprowadzenie eksperymentów oraz jednoczesne przeprowadzanie wielu eksperymentów przez różnych użytkowników.

Przed przystąpieniem do eksperymentów z daną aplikacją należy załadować do wstrzykiwacza jej binaria oraz dane wejściowe. Wówczas system przeprowadza tak zwany przebieg wzorcowy. Jest to wykonanie badanego programu w niezakłóconym środowisku, w celu zbadania jego normalnej pracy i określenia poprawnego wyniku dla zadanych danych wejściowych. W czasie tego wykonania zbierane są statystyki dotyczące instrukcji maszynowych, z których składa się program i poszczególne moduły. Ładując program określamy takie atrybuty, jak: nazwa, wersja oraz kompilator, którym skompilowano program.

Zlecając nowy eksperyment należy wybrać program i przebieg wzorcowy, do którego będzie odwoływał się eksperyment oraz podać parametry wstrzykiwanych błędów. Każdy eksperyment składa się z wielu testów – pojedynczych wykonań programu z zakłóceniem. System zbiera szczegółowe dane o wstrzykniętym zakłóceniu i jego wpływie na zachowanie aplikacji. Na podstawie porównania wyniku pracy aplikacji w teście z przebiegiem wzorcowym określana jest jego poprawność. Wyróżnione są cztery wartości: COR - aplikacja zakończyła działanie poprawnie i zwróciła poprawny wynik; INC - aplikacja zakończyła działanie poprawnie, ale zwróciła niepoprawny wynik; SYS - nieobsłużony w aplikacji wyjątek spowodował przerwanie działania aplikacji oraz TMO - system operacyjny przerywa działanie aplikacji po upływie określonego czasu bezczynności. Zarówno wynik przebiegu wzorcowego, jak i poszczególnych testów zapisywane są do plików tekstowych. Następnie system (poprzez procedury wbudowane) parsuje je i uzupełnia strukturę relacyjnej bazy danych.

3. Hurtownia danych FEARS

System FEARS (*Fault Effects Analysis and Reporting System*) [4] umożliwia sprawne raportowanie i analizę wpływu błędów dla eksperymentów symulacyjnych. Składa się on z hurtowni danych z wielowymiarową kostką analityczną oraz modułów raportujących.

Hurtownia zbiera dane z trzech baz danych: konfiguracji eksperymentów, statystyk z przebiegów wzorcowych oraz wyników testów. Dane w tych trzech bazach są ze sobą powiązane: każdy test jest wykonywany w ramach jakiegoś eksperymentu, a eksperyment związany jest z jednym z przebiegów wzorcowych. Ponieważ dane są składowane w różnych bazach danych, relacje między nimi nie są wprost widoczne. Uzupełnieniem tych relacji, transformacją danych do docelowego modelu hurtowni (opartego na schemacie gwiazdy) oraz wyliczaniem dodatkowych atrybutów zajmuje się proces ETL (*Extract, Transform, Load*). Ostatnim etapem jest budowa kostki analitycznej, w której wymiary są atrybutami dyskretnymi i porządkowymi, w tabeli faktów zaś są atrybuty liczbowe. Pojedynczy rekord tabeli faktów reprezentuje jeden test. W sumie dane z tabeli faktów i kilkadziesiąt miar wyliczanych w kostce mogą być rozważane w kontekście 17 hierarchii wymiarów.

Hurtownia zawiera także 37 wyliczeń opisujących każdy test. Są one tworzone w trakcie procesu ładowania danych do hurtowni. Można z nich monitorować: zachowanie aplikacji po wstrzyknięciu błędu, zmiany kodu lub adresu instrukcji po wystąpieniu zakłócenia czy też reakcję aplikacji na pojawiające się po błędzie wyjątki. Na danych wielowymiarowych i tabelach relacyjnych przygotowano wiele raportów, które mogą służyć wstępnej ich analizie. Pierwsza wersja systemu FEARS była zbudowana na podstawie oprogramowania SAS 9.1. Obecnie hurtownia i raporty działają w środowisku Microsoft SQL Server 2008. Warto zaznaczyć, że w przeciwieństwie do hurtowni danych budowanych do zastosowań biznesowych, w FEARS nie ma wymiaru czasu (moment wykonania testu nie wpływa na jego wynik). Występują tu jednak relacje następstwa czasowego, na przykład drugi wyjątek po zakłóceniu wystąpi zawsze później niż pierwszy i wcześniej niż trzeci.

4. Odkrywanie wiedzy w danych z hurtowni

Celem zastosowania metod eksploracji danych do analizy wyników eksperymentów symulacji błędów jest wsparcie odkrywania zależności, które mają wpływ na wiarygodność aplikacji. Do tego zadania najlepszy będzie model klasyfikatora opartego na regułach. Klasyfikator ten przyporządkuje każdy test do jednej z czterech klas reprezentujących możliwe wyniki testu: COR, INC, SYS, TMO. Następnie z reguł klasyfikujących będziemy wnioskować o zależnościach w danych. Proces ten powinien składać się z następujących etapów: przygotowanie danych, wybór istotnych atrybutów, wybór i budowa modelu analitycznego oraz ocena jakościowa modelu i analiza otrzymanych wyników.

4.1. Przygotowanie danych

Proces ładowania danych do hurtowni przetwarza je do docelowej struktury wielowymiarowej i tworzy wyliczenia. Następnie uzupełniane są atrybuty, których wartości nie można określić na podstawie danych zebranych przez symulator. Jeśli symulator nie zebrał właściwych danych, może to wynikać z jednej z poniższych przyczyn:

- w skutek błędu aplikacja testowa przestała działać przed zebraniem danych (np. wyjątek systemowy zatrzymał aplikacje przed wykonaniem 10 instrukcji po wstrzyknięciu błędu);
- zjawisko opisywane przez atrybut nie miało miejsca w trakcie testu (np. nie został wygenerowany żaden wyjątek).

W takich sytuacjach wartości puste niosą ze sobą pewną informacje o teście. W zależności od atrybutu stosujemy jedną z poniższych strategii radzenia sobie z brakami:

- Uznanie braku wartości za jedną z możliwych wartości atrybutu – wówczas dobrze jest zastąpić wartość *null* w bazie danych stałą, oznaczającą brak wartości.

- Pominięcie przykładów z brakującymi wartościami atrybutów.
- Nieużywanie atrybutu z dużą liczbą brakujących wartości w odkrywaniu wiedzy.

W klasycznym procesie budowy modelu, ostatnim elementem przygotowania jest podzielenie danych na zbiory trenujący i testowy. Ponieważ w przypadku analizy danych z symulatora nie zamierzamy stosować otrzymanego modelu do klasyfikacji, nie potrzebujemy osobnego zbioru danych testowych do oceny modelu. Nie należy obawiać się zbyt dopasowanego modelu, ponieważ im lepiej dopasowany model, tym lepiej opisuje on zależności w danych, które pragniemy odkryć.

Przy budowie zbioru trenującego należy wziąć pod uwagę rozkład niektórych ważnych atrybutów w danych. Na przykład *a priori* wiemy, że dla różnych grup lokalizacji wstrzykiwanego błędu (np.: pamięć kodu lub danych, rejestry procesora) większość aplikacji wykazuje różną wrażliwość. Dlatego należy przygotować zbiór trenujący z podobną liczbą testów dla każdej lokalizacji. Można to uzyskać przez wykluczenie losowo wybranych testów dla przeważającej lokalizacji lub rozmnożenie testów dla słabiej reprezentowanej lokalizacji.

4.2. Wybór atrybutów

Na podstawie pochodzenia atrybutów, możemy podzielić je na 3 grupy:

- Konfiguracja eksperymentu i informacje o miejscu wstrzyknięcia błędu. Zależności wśród tych atrybutów są znane, ponieważ wynikają z budowy wstrzykiwacza.
- Informacje o zachowaniu się aplikacji po wstrzyknięciu błędu. Te atrybuty zależą od specyfiki aplikacji i konfiguracji testu.
- Atrybuty wyliczane na podstawie dwóch wcześniejszych grup przy ładowaniu danych do hurtowni. Zależą od atrybutów z pozostałych grup i od siebie nawzajem. Część tych zależności wynika z algorytmu wyliczania atrybutu.

Atrybuty z różnych grup mają różną naturę, Dodatkowo, między grupami występują skomplikowane zależności. W związku z tym, model zbudowany na podstawie atrybutów z różnych gryp byłby bardzo trudny w analizie. Dlatego należy wybrać jedną z nich.

Przy wyborze atrybutów należy także wziąć pod uwagę do jakich klas będziemy przyporządkowywać testy. Część atrybutów nie ma znaczenia przy rozpatrywaniu przyczyn zakończenia testu z pewnymi wynikami. Na przykład o zakończeniu testu z wynikiem COR lub INC nie możemy wnioskować na podstawie informacji o kodzie i miejscu pojawienia sie wyjątku terminującego, gdyż występuje on tylko dla zakończenia SYS. Do budowy modelu należy wybrać takie atrybuty, z których będzie możliwe wnioskowanie o wszystkich klasach, wymienionych na początku rozdziału 4. Spośród dostępnych grup atrybutów taką cechę posiadają tylko atrybuty opisujące konfigurację eksperymentu i miejsce wstrzyknięcia błędu. Dlatego one zostały użyte do budowy pierwszego modelu w systemie FEARS.

W wybranej grupie znajduje się 27 atrybutów, o różnej liczbie możliwych wartości słownikowych lub liczbowych. W celu przyśpieszenia procesu budowy docelowego modelu klasyfikatora, należy ograniczyć liczbę atrybutów. W pierwszym etapie należy wyeliminować atrybuty, które dla wybranego zbioru testów są niezmienne w danych trenujących. Na podstawie pozostałych atrybutów budujemy model grupujący testy ze zbioru trenującego. Grupowania dokonujemy za pomocą algorytmu *Microsoft Expectation Maximization* [9]. Budujemy wiele modeli grupowania, z rożnymi parametrami konfiguracyjnymi i wybieramy najlepszy. Oceny otrzymanych modeli dokonujemy na dwóch płaszczyznach. W otrzymanych grupach badamy rozkład atrybutu decyzyjnego - wyniku testu. Następnie w kolejnych grupach porównujemy rozkład wyniku testu z rozkładem w całym zbiorze i w innych grupach. Jeśli rozkład tego atrybutu w grupach jest znacząco różny od rozkładu w całej populacji, przyjmujemy, że atrybuty, które są wyróżnikami grup będą także istotne w klasyfikacji do klas będących zakończeniami testów. Tak więc, im bardziej różnorodny rozkład, tym lepszy model. Sprawdzamy także, jak dany model grupowania sprawdzi się jako klasyfikator przyporządkowujący testy do wyników. Ocena grupowania jako klasyfikatora zostanie wykonana na podstawie wykresu *Lift* [9]. Wykres *Lift* przedstawia w sposób graficzny skuteczność klasyfikatora. Na osi odciętych wykresu oznaczono procent przypadków testowych, które zostały już poddane klasyfikacji, na osi rzędnych zaś procent przypadków poprawnie zaklasyfikowanych. Wykres dla idealnego modelu będzie prostą o równaniu $y = x$. Ostatecznego wyboru atrybutów dokonujemy na podstawie najlepszego podziału na grupy. Dla każdej z powstałych w nim grup wybieramy 6 atrybutów, których wartości najbardziej odróżniają daną grupę od całego zbioru trenującego. Do budowy klasyfikatora zostaną użyte wszystkie atrybuty, które pojawiły się dla co najmniej dwóch grup. Dodatkowo, do zbioru wybranych atrybutów mogą być dołączone atrybuty opisujące testowany program, na przykład: kompilator oraz wersja kodu. Oceny metody wyboru atrybutów dokonamy przez porównanie klasyfikatorów zbudowanych na atrybutach wybranych i na atrybutach użytych w grupowaniu.

4.3. Wybór algorytmu i budowa modelu

Zadanie klasyfikacji może być realizowane przez różne algorytmy [10]. Ponieważ jednak chcemy z modelu wydobyć wiedzę o zależnościach w danych, należy wybrać taki klasyfikator, który ma reprezentację zrozumiałą dla człowieka. Z tego powodu należy wykluczyć modele takie, jak naiwny klasyfikator Bayes'a oraz skomplikowane modele, zbudowane przez złożenie kilku algorytmów lub modeli (wiedza o zależnościach w danych jest w tych modelach „rozproszona" na poszczególne składowe i nie jest łatwa do przełożenia na zależności w danych). Istotnym aspektem jest także nakład pracy, jakiego wymagało zastosowanie algorytmu. Z tego powodu w pierwszej kolejności rozważane były algorytmy dostępne w środowisku *Microsoft SQL Server 2008 Analysis Services* (SSAS) [9]. Z algorytmów udostępnia-

nych przez to narzędzie postawione wymagania najlepiej spełniają dwa: drzewa decyzyjne oraz reguły asocjacyjne. W początkowej fazie badań zbudowano modele oparte na tych algorytmach. Model oparty na drzewie decyzyjnym był bardziej skomplikowany. Otrzymane z niego reguły miały do 16 warunków w poprzedniku. Część z tych warunków była redundantna. Natomiast otrzymane reguły asocjacyjne były prostsze. Za regułami asocjacyjnymi przemawia także aspekt nowości. Drzewa decyzyjne były już używane do analizy danych z symulatora błędów [7]. Chociaż budowa klasyfikatora nie jest pierwotnym zastosowaniem reguł asocjacyjnych, w literaturze można znaleźć przykłady takiego ich zastosowanie [10]. Oryginalnie algorytm ten tworzy reguły z dowolnymi atrybutami w następniku, jednak SSAS [9] pozwala tak go sparametryzować, aby tworzył jedynie reguły posiadające konkretny atrybut w następniku. Ograniczony zostanie także maksymalny rozmiar zbiorów częstych do 4.

4.4. Ocena jakościowa modelu i analiza otrzymanych wyników

Oceny otrzymanego modelu należy dokonać na dwóch płaszczyznach: oceny klasyfikatora, zbudowanego na podstawie otrzymanych reguł oraz oceny poszczególnych reguł. Ocena klasyfikatora to sprawdzenie, czy klasyfikator poprawnie przyporządkowuje poszczególne testy do możliwych klas. W tym celu wykorzystana zostanie macierz pomyłek klasyfikatora i wykres *Lift* (patrz rozdział 4.2). Ocena modelu będzie także dokonana przez porównanie z klasyfikatorem zbudowanym na tych samych danych i atrybutach, ale z użyciem innego algorytmu (np. z naiwnym klasyfikatorem Bayes'a).

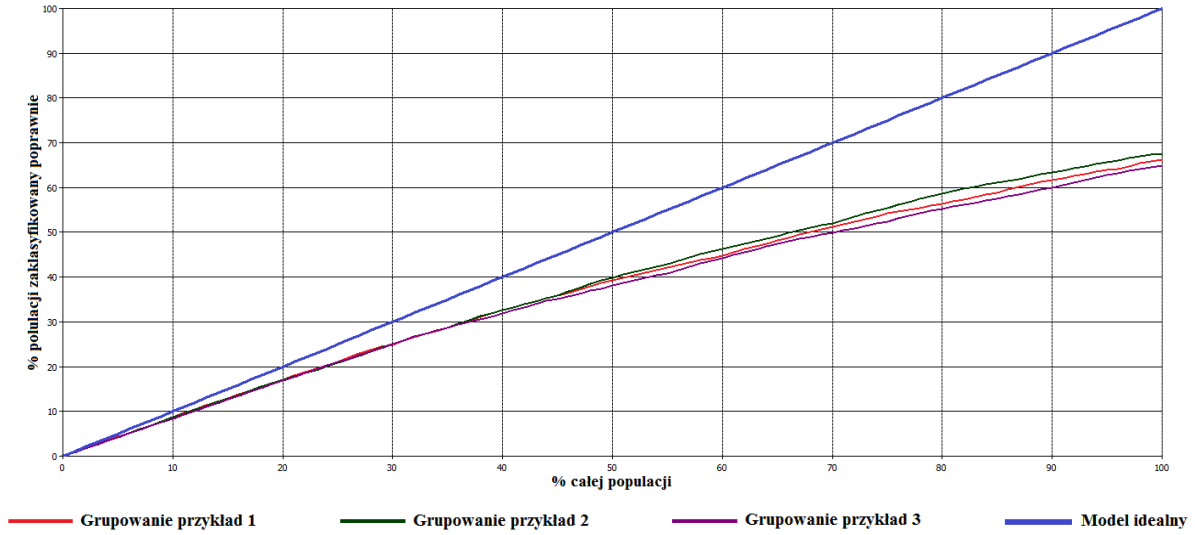
Oceny poszczególnych reguł dokonujemy łącznie z wyborem reguł, które zostaną poddane analizie. Podstawą oceny reguł będą: prawdopodobieństwo reguły oraz korelacja między jej następnikiem i poprzednikiem. Prawdopodobieństwo będzie liczone jako iloraz wsparcia reguły i wsparcia jej poprzednika. Korelacja wyznaczana jest jako logarytm ilorazu ufności reguły i względnego wsparcia jej następnika. Korelacja równa zero oznacza zdarzenia niezależne, mniejsza od zera zdarzenie skorelowane negatywnie, a większa od zera zdarzenia skorelowane pozytywnie. Dodatkowym kryterium może być względne wsparcie reguły. Warto zaznaczyć, że interesujące mogą być nie tylko pewne reguły o dużym prawdopodobieństwie i wsparciu. Dlatego analizie będą poddawane reguły o dużym prawdopodobieństwie, ale także o skrajnych wartościach korelacji. W pierwszej kolejności analizie zostaną poddane reguły o następniku innym niż COR, ponieważ bardziej interesujące są czynniki powodujące błędy z aplikacji. Wraz z wybranymi regułami analizowane będą także reguły o poprzedniku takim, jak w wybranej regule, ale innym następniku.

5. Eksperymenty

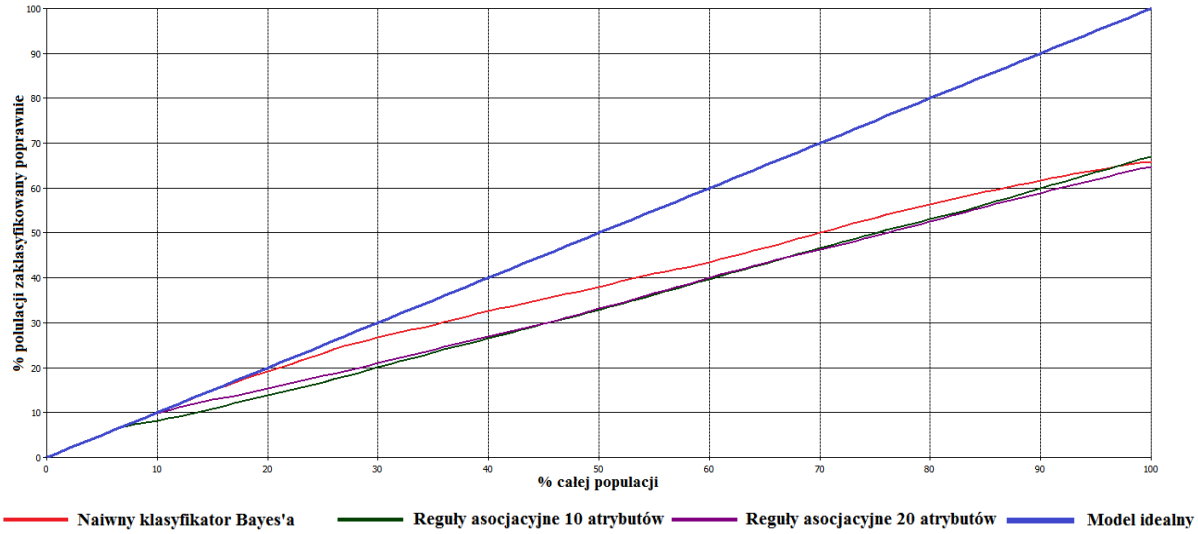
Zastosowanie opisanej powyżej metody analizy zostanie przedstawione na przykładzie danych pochodzących z eksperymentów wstrzykiwania błędów w sterownik reaktora chemicznego. Celem badanej aplikacji jest kontrola procesu w reaktorze za pomocą dwóch zmiennych (odpowiadających dopływowi składników do reaktora), tak aby uzyskać odpowiednie parametry wynikowego produktu. Wcześniejsze badania nad niezawodnością tej aplikacji metodami symulacyjnymi zostały przedstawione w pracach [3, 4]. W hurtowni danych zgromadzono wyniki eksperymentów dla 6 wersji kodu sterownika (każda z kolejnych wersji od 1 do 6 jest wyposażona w coraz bardziej skomplikowane mechanizmy wykrywania i tolerowania bledów), skompilowanych kompilatorem VC++ (*Microsoft Visual C++*) w wersjach 6.0 oraz 2005, co daje w sumie 12 wersji aplikacji. W aplikację wprowadzano zakłócenia polegające na pojedynczej inwersji pojedynczego bitu, z użyciem 2 strategii wyboru momentu zakłócenia. Badano 3 lokalizacje: rejestry procesora oraz kod aktualnie wykonywanej instrukcji w pamięci podręcznej lub *RAM*. Zbiór danych trenujących powinien składać się z podobnej liczby testów dla każdej z 2 zastosowanych strategii wyboru momentu wstrzyknięcia błędu, dla każdej z 3 lokalizacji błędu oraz dla każdej z 12 wersji aplikacji (patrz rozdział 4.1). Ponieważ w sumie wykonano 29 eksperymentów od 1 000 do 100 000 testów każdy, dla niektórych kombinacji atrybutów (strategia, lokalizacja, wersja) mamy 1 000 testów dla innych ponad 100 razy więcej. W związku z tym podjęto decyzje, że należy przygotować zbiór danych trenujących, zawierający po ok. 10 000 testów dla każdej z 72 kombinacji. Dla kombinacji o większej liczbie testów ograniczono zbiór użytych eksperymentów, aby uzyskać odpowiednią liczbę testów, a dla tych, gdzie było zaledwie 1 000 testów, każdy test został dołączony do zbioru testowego dziesięciokrotnie.

W pierwszym etapie wyboru atrybutów wyeliminowano 7 atrybutów, które dla wybranego zbioru testów są niezmienne. Następnie pozostałe 20 atrybutów poddano analizie z wykorzystaniem grupowania. Rysunek 1 przedstawia wykresy *Lift* dla 3 wybranych modeli. Widać na nim, że model 2 jest nieznacznie lepszym klasyfikatorem. Ten model został wykorzystany do wyboru atrybutów. Ostatecznie wybrano10 atrybutów.

Na podstawie wybranych atrybutów zbudowano 2 klasyfikatory: oparty na regułach asocjacyjnych do wnioskowania o zależnościach w danych oraz naiwny klasyfikator Bayes'a do oceny porównawczej. Do oceny skuteczności metody wyboru istotnych atrybutów zbudowano model opierający się na regułach asocjacyjnych na bazie 20 atrybutów, których używano w grupowaniu. Jak pokazuje wykres na rys. 2, model oparty na regułach asocjacyjnych i przygotowany na 10 atrybutach jest lepszym klasyfikatorem niż reguły asocjacyjne przygotowane na 20 atrybutach oraz niż naiwny klasyfikator Bayes'a. Analiza macierzy pomyłek dla poszczególnych modeli prowadzi do podobnych wniosków.



Rys. 1. Wykres Lift dla przykładowych modeli grupowania
Fig. 1. Lift chart for example clustering models



Rys. 2. Wykres *Lift* dla przygotowanych klasyfikatorów
Fig. 2. Lift chart for prepared classificators

Następnie dokonano oceny reguł i wyboru, które z nich zostaną przeanalizowane. Gotowy model składa się z 16 831 reguł, z czego 961 reguł miało prawdopodobieństwo równe jeden: 8 z nich w następniku miało wynik testu SYS i korelację 0,677, pozostałe COR i korelacje od 0,189 do 0,192. W zbiorze reguł jest:

- 8 764 reguły o następniku COR i prawdopodobieństwie >= 0,5, korelacja od -0,176 do 0,208;
- 54 reguł o następniku INC i prawdopodobieństwie od 0,5 do 0.6, korelacja od 0,556 do 0,675, (wszystkich 2 281, także z korelacją ujemną i z korelacją > 1);
- 503 reguł o następniku SYS i prawdopodobieństwie >= 0,5 (wszystkich 4 479);

- 0 reguł o następniku TMO i prawdopodobieństwie >= 0,5 – to zakończenie testu jest bardzo słabo reprezentowane i takich reguł w ogóle nie ma.

Spośród reguł o prawdopodobieństwie >= 0,5 analizie zostały poddane te o największej i najmniejszej korelacji. 25 reguł ma korelację > 1, ale ich prawdopodobieństwo jest nieduże (<= 0,369). Wszystkie z tych reguł mają w następniku INC. Mimo małego prawdopodobieństwa, warto przyjrzeć się im bliżej, bo mogą świadczyć o jakiejś anomalii w danych. Za analizą tych reguł przemawia także stosunkowo duże względne wsparcie (> 0,12).

W pierwszej kolejności analizie zostanie poddane 8 reguł o prawdopodobieństwie 1, korelacji 0,677 i zakończeniu testu SYS. Są to (kolejno atrybuty: lokalizacja wstrzykniętego błędu, kompilator, wersja aplikacji):

- Rejestr_ESP i VC++_6.0 i v.1 -> SYS
- Rejestr_ESP i VC++_6.0 i v.2 -> SYS
- Rejestr_ESP i VC++_6.0 i v.3 -> SYS
- Rejestr_ESP i VC++_6.0 i v.4 -> SYS

Są to także cztery równoważne reguły, zawierające w poprzedniku atrybut nazwę zasobu, na którym operuje zakłócana instrukcja (ESP) zamiast lokalizacji wstrzyknięcia. Przytoczone reguły nie dziwią, ponieważ z poprzednich badań wynika, że zastosowane w wersjach aplikacji 2-4 sprzętowe wykrywanie wyjątków nie działa w kompilatorze VC++ 6.0 [4]. Tak więc te wersje, a także wersja 1, pozbawiona jakichkolwiek mechanizmów wykrywania i tolerowania błędów, nie są zabezpieczone przed wyjątkami, które powoduje zmiana wskaźnika stosu (ESP). Dziwi jednak, że zaobserwowana własność nie występuje dla 1 wersji aplikacji i kompilatora VC++ 2005. Zapewne wynika to z jakiś szczególnych różnic w budowie tych dwóch kompilatorów i wymaga sprawdzenia w dokumentacji. Jednak już na podstawie posiadanej wiedzy możemy wysnuć wniosek, że aplikacje skompilowane kompilatorem VC++ 2005 są bardziej odporne na błędy.

Następnie analizie poddana zostanie reguła o najwyższej korelacji i 2 reguły o tym samym poprzedniku co ona. Są to (w poprzedniku zawierają atrybut nazwę zasobu, na którym operuje zakłócana instrukcja):

- Brak_zasobu -> COR o prawdopodobieństwie = 0,580 i korelacji = -0,158;
- Brak_zasobu -> INC o prawdopodobieństwie = 0,191 i korelacji = 2,698;
- Brak_zasobu -> SYS o prawdopodobieństwie = 0,228 i korelacji = 0,139.

Wartość atrybutu Brak_zasobu oznacza, że błąd został wstrzyknięty w instrukcje bez argumentów. Wysoka pozytywna korelacja przy małym prawdopodobieństwie świadczy, że takie instrukcje są bardziej podatne na błędy, które spowodują zmianę wyniku pracy programu, ale mimo to błędy te są rzadkie. Analiza innych reguł z tą samą parą atrybut – wartość w poprzedniku pomoże określić, w jakich przypadkach to występuje. 10 z tych reguł ma INC w następniku oraz korelację > 1 i prawdopodobieństwo >= 0,369. Wynika z nich, że zakłóce-

nie aktualnie wykonywanej instrukcji w pamięci RAM dla instrukcji bez argumentów częściej wywołuje zwrócenie przez aplikację niepoprawnego wyniku. Analizując kolejne reguły można zauważyć, że najbardziej wrażliwe są instrukcje jednostki zmienno pozycyjnej, z których głównie składa się badana aplikacja. Sugeruje to, że czynnikiem mającym największy wpływ na niezawodność badanej aplikacji są instrukcje zmiennopozycyjne i w tym miejscu należy wprowadzać kolejne ulepszenia niezawodności.

6. Podsumowanie

W niniejszym artykule zaproponowano metodykę analizy danych z symulatorów błędów, opartą na metodach odkrywania wiedzy. Jej celem jest wsparcie poszukiwanie cech badanej aplikacji, które mają szczególny wpływ na jej niezawodność. Pozwala ona na zaoszczędzenie czasu, potrzebnego by przejrzeć wiele raportów i obejrzeć dane w wielu wymiarach w poszukiwaniu wrażliwych miejsc aplikacji. Jak pokazano, zaproponowana metodyka spełnia to zadanie. Metodyka ta może być szczególnie przydatna, gdy testom poddawane są aplikacje dobrze nieznane przeprowadzającym eksperymenty.

W kolejnych etapach pracy należy sprawdzić zaproponowaną metodykę na innych aplikacjach testowych. Warto także wzbogacić ją o nowe algorytmy klasyfikacji oraz nowe metody wyboru istotnych atrybutów. Wśród algorytmów klasyfikacji warto rozważyć, bliskie regułom asocjacyjnym, pewne i ogólne reguły decyzyjne. Warto także podjąć pracę nad bardziej efektywną i szybszą metodą oceny i wyboru reguł do dalszej analizy.

Innym aspektem, który powinien być zbadany w przyszłości są zależności między różnymi grupami atrybutów. Należy rozważyć budowę modelu wnioskującego na podstawie atrybutów z grupy pierwszej (lub pierwszej i drugiej) o atrybutach z grupy drugiej (lub trzeciej). W ten sposób można na przykład wykryć czynniki wpływające na zakończenie się testu konkretnym wyjątkiem.

BIBLIOGRAFIA

1. Benso A., Prinetto P.: Fault injection techniques and tools for embedded systems reliability evaluation. Kluwer Academic Publisher 2003.
2. Gawkowski P., Sosnowski J.: Developing Fault Injection Environment for Complex Experiments. In: 14th IEEE International On-Line Testing Symposium, 2008, s. 179÷181.
3. Gawkowski P., Ławryńczuk M., Marusak P. M., Tatjewski P., Sosnowski J.: On improving dependability of the numerical GPC algorithm. [in:] European Control Conference, Budapest, Hungary 2009, s. 1377÷1382.

4. Gawkowski P., Kuczyńska M. A., Komorowska A.: Fault Effects Analysis and Reporting System for Dependability Evaluation. Rough Sets and Current Trends in Computing (Szczuka M.). LNAI, Vol. 6086, Springer, Heidelberg 2010, s. 524÷533.
5. Parsa S., Vahidi-Asl M., Naree S. A.: Finding Causes of Software Failure Using Ridge Regression and Association Rule Generation Methods. In: 9th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD '08), 2008, s. 873÷878.
6. Pintér G., Madeira H., Vieira M., Majzik I., Pataricza A.: Integration of OLAP and data mining for analysis of results from dependability evaluation experiments. International Journal of Knowledge Management Studies, Vol. 2, 2008, s. 480÷498.
7. Sosnowski J., Zygulski P., Gawkowski P.: Developing data warehouse for simulation experiments. Rough Sets and Emerging Intelligent Systems Paradigms (Kryszkiewicz M.). LNAI, Vol. 4585, Springer, Heidelberg 2007, s. 543÷552.
8. Sosnowski J., Tymoczko A., Gawkowski P.: An approach to distributed fault injection experiments. 7th International Conference on Parallel Processing and Applied Mathematics (Wyrzykowski R). LNCS, Vol. 4967, Springer, Heidelberg 2008, s. 361÷370.
9. SQL Server Books Online: http://msdn.microsoft.com/en-us/library/ms130214.aspx.
10. Han J., Kamber M.: Data mining, concepts and techniques, Morgan Kaufman Pub, 2005.

Recenzenci: Dr inż. Marek Sikora
Prof. dr hab. inż. Alicja Wakulicz-Deja

Wpłynęło do Redakcji 15 stycznia 2011 r.

Abstract

Article describes a methodology of analysis of data from faults simulator. The methodology is based on association rules. The goal of the analysis is to discover application's features important for its dependability. Faults simulation experiments produce huge amount of data, which should be analyzed mostly automatically.

Data from experiments are stored in data warehouse with reporting and data mining capabilities. Knowledge discovery consists of following steps: data preparation, choosing important attributes (based on expert knowledge and using clustering algorithm), construction of data mining model (based on association rules), model evaluation, results analysis and interpretation. Usefulness of proposed methodology is shown on an example of analysis of data from fault injection experiments for chemical reactor controller.

Adres

Agnieszka KOMOROWSKA: Politechnika Warszawska, Instytut Informatyki,
ul. Nowowiejska 15/19, 00-665 Warszawa, Polska, A.Komorowska@ii.pw.edu.pl.